REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
2/20/2006

2. REPORT TYPE
Final

3. DATES COVERED *(From - To)*
7/01/01 - 12/31/04

4. TITLE AND SUBTITLE
Understanding Mobile Code and Secure Execution Environments

5a. CONTRACT NUMBER
N000014-01-1-0981

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Keith Marzullo and Bennet Yee

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Regents of the University of California, San Diego
9500 Gilman Drive
La Jolla, CA 92093 - 0934

8. PERFORMING ORGANIZATION REPORT NUMBER
04PR04452-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research

10. SPONSOR/MONITOR'S ACRONYM(S)
ONR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
This final report has no restrictions on distribution.

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

13. SUPPLEMENTARY NOTES
Professor Bennet Yee was the originating P.I. on this contract. Professor Keith Marzullo assumed control for the last year when Professor Yee left UCSD.

14. ABSTRACT
The Sanctuary project investigated the engineering fundamentals of mobile code security. Under this project, we: (1) discovered and proved a simple way to implement symmetric key cryptographic schemes with the forwarding security property; (2) discovered and investigated the technique of secure speculative predictive remote execution. Also investigated were issues in fault tolerance for a specific problem domain: Grid Services. Six M.S. degrees were awarded under this project, and one Ph.D. degree under progress has material that was produced under this contract.

15. SUBJECT TERMS
Mobile code security, mobile code, mobile agents, system security, fault tolerance.

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
U	U	U	UU	7	Keith Marzullo
					19b. TELEPHONE NUMBER *(Include area code)* 858 534-3729

ONR Award N00014-01-1-0981
July 2001 — December 2004
Understanding Mobile Code and Secure Execution Environments
Final Report

Keith Marzullo Bennet Yee

February 21, 2006

1 The Sanctuary Project

The Sanctuary project investigated the engineering fundamentals of mobile code security. We believe that mobile code will become a more attractive component of a distributed system designer's toolbox as an inevitable result of technology trends. Hardware trends continue to make computational resources ever cheaper and more plentiful, while the constraints of geographic separation remain essentially fixed. Long-distance communication latencies will become a critical bottleneck. Co-locating code and resources reduces communication latency and increases overall system throughput.

Often, the intended goal of using mobile code—to make distributed systems operate more efficiently—seems to become unreachable once the realities of widely distributed systems is considered. The computation resources in these systems belong to disparate security domains, and trusting that the remote execution of some code occurred correctly requires a tremendous leap of faith.

1.1 Goals

The goals of the Sanctuary project is to investigate the security issues that arise when mobile code is used as a component of a distributed system.

The security problem examined decomposes into two main areas: (1) server security and (2) mobile code (or mobile agent) security. For the former, the assets being protected are the resources accessible at a host for mobile code, as well as resources carried by mobile agents from one user or principle that should be partitioned from access by another. The host should provide fine-grained controlled access to these resources, and of course provide the necessary basic support for mobile code.

The latter problem is more fundamental in nature and variations of it has been explored earlier as *copy protection* [11], *trusted computing* [6, 25], or the *secure remote execution* problem [3, 7]. This can be subdivided into two security properties for computation analogous to that for communication security: (1) computational confidentiality, and (2) computational integrity. The former refers to being able to compute a function remotely without the owner of the remote machine hosting the computation being able to determine what that remote code is computing. The latter refers to being able to trust that the result of the remotely executing code had not been tampered with, that the result would be identical if the computation had occurred on a trusted secure machine.

2 Accomplishments

The Sanctuary project's investigation in mobile code security resulted in advances in several areas.

We discovered and proved correct, in the concrete security setting, a simple way to implemented symmetric key cryptographic schemes with the *forward security* property [23, 4]. By using forward-secure cryptography, we enabled multi-hop mobile agents to commit to partial results determined prior to visiting any malicious hosts, thereby making it computationally impossible for those malicious hosts to tamper with or otherwise falsify the earlier partial results.

We investigated techniques to factor out support for process migration from the mobile code system, enabling the use of a simpler, more modular architecture.

We discovered and investigated techniques to monitor remotely executing processes, as well as *secure speculative/predictive remote execution* (S^2PRE), enabling the use of mobile code on remote, untrusted machines to improve distributed system throughput without compromising computational integrity [24]. While this technique does not provide any computational confidentiality, it allows us to use mobile code to speed up remote procedure calls (RPCs) so that the RPCs appear to have the same latency as with full resource co-location.

2.1 Mobile Code Execution Environments

We expended a great deal of effort in designing and building a secure mobile code execution environment. Here, the primary goal was to ensure that rogue mobile code cannot compromise the security of the hosting service. Doing this enabled us to examine closely the interrelationship between mobile code security and server security, and to build in support within the server needed to make the implementation of mobile code security techniques feasible.

Matthew Hohlfeld designed and implemented the Sanctuary server, a secure mobile code execution environment. He focussed primarily on server security but also on providing the tools/hooks needed to address agent security [8].

At first approximation, the mobile code server is essentially a Java-based sandbox that controls access to local resources. It provides a rich interface that goes far beyond simply controlling local file or network access. The server provides interfaces that enable local service to use a secure inter-agent communication mechanism to authenticate, authorize, and account for use of access-restricted resources. For specialized resources such as cryptographic keys that mobile code might carry, it also provides protocol coordination to ensure that process migration is coordinated with securely transfering cryptographic keys.

Another class of resource that is carried by migrating processes is communication endpoints. This work was the core of Juliana Wong's Masters thesis. She designed and built a secure network interprocess communication system (IPC) for mobile agents [21], patterned largely on the IPC mechanism of the Mach operating system [2].

In our mobile code system, the mobile agents carry with them access rights to ports, which can be thought of as message queues. Associated with each port is exactly one extant receive right but there may be many send rights as well as send-once rights. We apply a simple cryptographic technique—key splitting—to associate unique cryptographic keys with each send or send-once right; this enables us to apply message authentication codes to messages so that not only would a host compromise have to occur before forged messages may be created, but agents (or their hosts) that receive a send right could only forge messages to appear to originate from a source below it in the derivation tree, i.e., a send (or send-once) right that derived from the send right held by the compromised agent (or host).

The expected frequency at which mobile agents migrate has a marked impact on the design of our distributed mobile-agent IPC mechanism. If the agent holding the receive rights to a port migrates, the (remote) agents holding send rights must either be notified of the new location of the port, or the host which the agent is leaving must remain available and be able to forward messages. In this work, such a host only temporarily forwards messages and can stop at any time. If a message arrives which should be forwarded, it is simultaneously forwarded and a path update is sent to the receiver this "forwarding path compression" updates frequent sender with the current location of port receive rights. If the forwarder has been rebooted or has garbage collected the forwarding information, it sends a failure message and the sender must use a "home agent server" (akin to that of a "home agent" in mobile IP) to enable the discovery of a port's current location should the location update fail.

2.2 Forward-Secure Cryptography

To enable secure mobile agents to detect tampering with partial results arrived at earlier servers, we use forward-secure symmetric-key cryptography [4] to generate forward-secure authentication codes. The use of forward-secure cryptography in practice is complicated by the need to integrate the key derivation and transfer with the task of mobile agent migration.

Aditya Ojha investigated how to support forward-secure cryptography in our Java-based mobile code framework in his Masters thesis work. He implemented a Java cryptography library using Java Native Interface (JNI) that provides strong forward security properties [13]. This is needed because the Java memory model makes it impossible to ensure that cryptographic keys can be securely deleted. By using JNI to access C code, mobile agents are assured that if they are migrating away from a uncompromised server, then all traces of cryptographic keys will have been deleted once they arrive at the new server.

Forward-secure cryptography support and its integration with the mobile agent server is critical to supporting secure mobile code.

2.3 Programming Language Support for Migrating Processes

Simplicity of design and modularization enables us to more clearly reason about security. Rather than trying to capture the state of a running Java program by changing the low-level guts of the Java Virtual Machine (JVM) implementation [15], we chose to relieve the mobile agent server from any such responsibility. Instead, we move the complexity outside of the JVM and use a special compiler for a special dialect of Java and library code to implement process migration.

Our Java dialect provides support for continuations as first-class objects and process checkpointing and migration is implemented as a library package on top of these continuations. (We also implemented user-level threads in Java using continuations; since our dialect do not support multithreaded agents, this is actually useful!) Rahul Lahoti designed and implemented a compiler that accepts this dialect and outputs standard Java source code as his Master's thesis work [9, 12]. Because the compilation process involves a series of simple, highly stylized parse tree transformations, the correctness of the compiler is amenable to code inspection and thorough testing.

We believe that this approach of factoring out migration from the virtual machine makes it far easier to secure the system. The interfaces to the agent server remains simple, and the compiler can be easily tested in a completely independent manner from that of the base system. Furthermore, this independence insulates us from changes to the JVM, so, for example, improvements to the byte code compiler is readily incorporated into the system.

2.4 Making Secure Migration Decisions

Having a simple mechanism to migrate an agent to a new server is perhaps a necessary condition for secure migration, but it is not a sufficient one. Clearly, in a large-scale distributed mobile code environment where multiple security domains are involved, knowing whether the agent *should* migrate to the new server is critical.

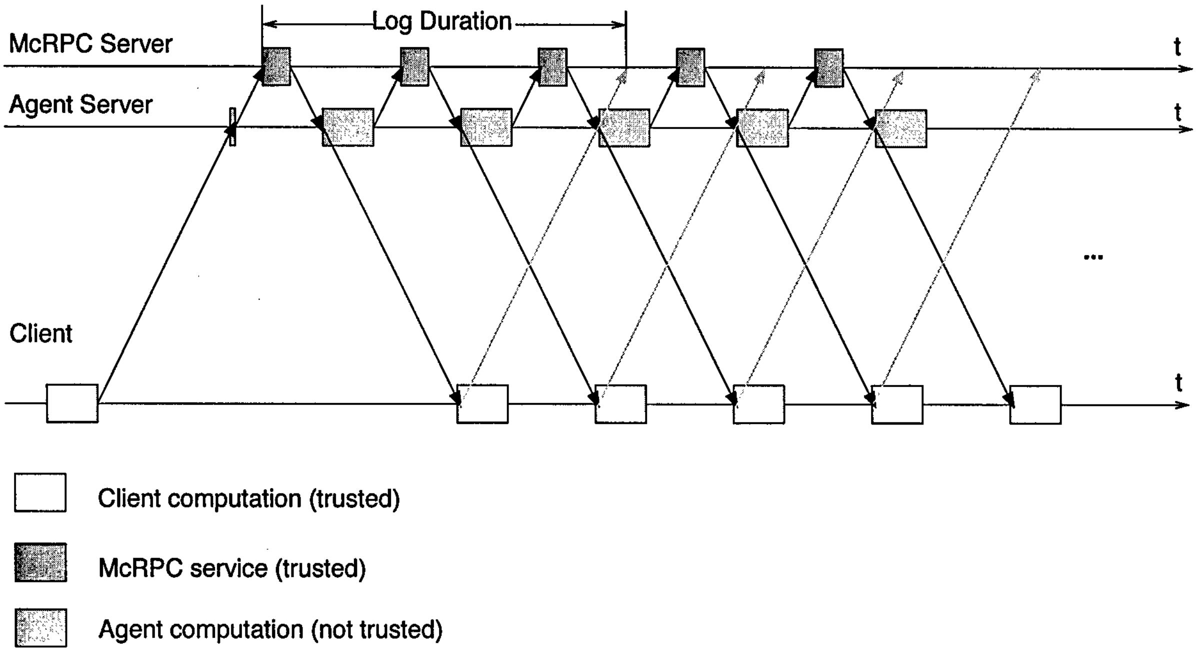
To this end, Poornaprajna Udupi investigated security attribute certificates and implemented software used to reason about trust as part of his thesis work [20]. In this work, we use attribute certificates to attest to security properties of hosts of mobile code. The security properties being attested to may range from the result of recent penetration tests, a system's Orange Book or Common Criteria evaluation [18] or FIPS 140-2 [19] rating, the amount of insurance / indemnification, etc. By explicating the security properties and permitting mobile code to make decisions based on this data, we enable the author and users of mobile code to define security policies to make explicit tradeoffs between the efficiency gains that would accrue from using a remote service and the amount of risk to undertake.

The security attribute certification infrastructure design is based on that of SDSI/SPKI [16, 5], and derives much from the simplicity and flexibility of the Lisp-like syntax. The extensibility implies having to deal with a wide range of attributes, including some that may change with time. The security attribute certification infrastructure includes an on-line attestation component to permit relying parties to ensure that a given attribute certificate has not been revoked.

2.5 Detecting Remote Tampering

More recently, we explored approaches that detects tampering with a remotely executing agent at times in its execution at other times than during migration or after the agent returns. The first approach, State Transition Inconsistency Detection (STID), uses code instrumentation to incorporate state-transition event sensors in the mobile code and allows a wide spectrum of performance versus sensitivity tradeoffs. The second approach, Secure Speculative/Predictive Remote Execution (S^2PRE), assumes the ability to rollback transactions and uses remote execution as a prediction unit co-located with remote resources to essentially eliminate long-distance RPC roundtrip latency. This scheme runs a program twice—once locally on trusted servers, and once, as a remote prediction unit, co-located with the remote resource—but is provably completely secure with respect to computational integrity.

As part of her Master's thesis work, Yekaterina Tsipenyuk investigated how to monitor remotely executing code to ensure that its integrity of execution is not compromised [17, 22]. We explored using static analysis techniques to identify code paths in which state transition sensors should be placed and to identify a set of state transitions that are *monotonic* in nature. When executed, the sensors transmit state transition events to a monitor process at a trusted server, and the monitor uses a state machine model to detect illegal state transitions. Katrina now works at Fortify Inc, a startup company that is commercializing using static analysis to detect security bugs in software.



To investigate S^2PRE, Scott O'Neil designed and implemented modifications and extensions to Sun's Remote Method Invocation (RMI) remote procedure call system [14]. On the first Mobile Code RPC (McRPC) to a server, we migrate a copy of the program to a mobile code server (the program's *shadow*) co-located with (or near) the RPC service and invoke the RPC from there. The RPC response and the RPC input parameters are signed by the RPC server and sent back to the mobile code server, which copies it to the original program. The original program verifies that the RPC input parameters are what it would have sent, accepts the RPC response by sending a "commit" message to the RPC server. Any future McRPCs will be take place in a lock-step manner: the shadow runs slightly ahead of the original and sends the RPC inputs to the RPC server, and by the original has computed the RPC inputs the forwarded RPC response will have arrived. Figure 2.5 shows how a McRPC is implemented using migration and normal RPCs.

Speculative execution using a redacted/simplified version of the program remains to be investigated. Using a redacted version of the program would trade off prediction accuracy with (perhaps) less information exposure—currently there is no computational confidentiality without applying other security techniques (e.g., [1]). Unfortunately, none of these other techniques are sufficiently efficient and would destroy the lock-step nature of S ^{2}PRE.

2.6 Support for Grid Services

Grid services were first developed in the context of wide-area scientific computation called grid computing. In many ways, grid computing was a return to the ideas of distributed computing that were studied ten or more years ago: a large number of moderately tightly coupled computation nodes providing a transparent high performance platform. The difference is that grid computing was doing this for real: the platform wasn't just meant for evaluation but is being used for production purposes. Hence, many issue that were glossed over are now vital. Some of the first issues that needed to be addressed was resource management.

At the same time that the computational grid was developing, web services were becoming widely deployed. Web services are widely used commercially to support web servers, which are now a commercially vital infrastructure. By growing out of web servers, web services did not directly address distributed computation. The idea of Grid services was thus created: to combine the grid computation and the Web services models. The goal is to create a platform that supports ambitious multisite applications (now often called "service oriented architecture", or SOA).

Important concerns for Grid services are security and fault tolerance. There are two characteristics of Grid services that impact how fault tolerance is best provided:

1) Grid services run on high level protocols. Being so high on the stack makes it hard to have short and tight

upper bounds on a process' responsiveness. This makes some simple and popular approaches based on perfect failure detectors, such as message logging and primary-backup protocols, unsuitable.

2) Grid services were first developed for resource scheduling. Grid scheduling often uses randomized algorithms to balance load more efficiently. Nondeterminism makes providing fault-tolerance harder.

Under this research grant, we launched research into mechanisms for services with the above characteristics. This work is the core of the PhD dissertation of Xianan Zhang (degree estimated in Fall 2006).

She's designed these mechanisms under some practical constraints: the performance overhead needed to be low; reusing existing grid mechanisms is highly desirable; adding fault tolerance to an existing grid serice should be simple - ideally, done without changing any code on the server.

This work was based on [10]. Work done under this grant is presented in [26] as well as three additional papers, two under review and one being prepared.

Publications Produced Under This Grant

- Mihir Bellare and Bennet Yee. Forward-security in private-key cryptography. In M. Joye, editor, *Topics in Cryptography—CT-RSA '03*, volume 2612 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.
- Matthew Hohlfeld, Aditya Ojha, and Bennet Yee. Security in the Sanctuary system. Technical Report CS2002-0731, Computer Science and Engineering Department, University of California at San Diego, December 2002.
- Matthew Hohlfeld and Bennet S. Yee. How to migrate agents. Technical Report CS98-588, Computer Science and Engineering Department, University of California at San Diego, La Jolla, CA, August 1998.
- Kjetil Jacobsen, Xianan Zhang, and Keith Marzullo. Group membership and wide-area master-worker computations. In *ICDCS 2003*, pages 570–581, Providence, RI, 2003.
- Rahul S. Lahoti. Continuations in Java. Master's thesis, University of California, San Diego, July 2003.
- Aditya Ojha. Tamper-evident mobile agents. Master's thesis, Computer Science and Engineering, University of California, San Diego, 2003.
- Scott Owen O'Neil. Secure mobile agent aware remote procedure calls in S^2PRE. Master's thesis, University of California, San Diego, 2004.
- Yekaterina Yevgenyevna Tsipenyuk. Detecting external agent replay and state modification attacks. Master's thesis, University of California, San Diego, 2004.
- Poornaprajna Venkatraj Udupi. Security attribute certification infrastructure (SACI). Master's thesis, University of California, San Diego, 2003.
- Juliana Wong. A secure network IPC system for mobile agent systems. Master's thesis, University of California, San Diego, 2004.
- Bennet Yee. Monotonicity and partial results protection for mobile agents. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS'03)*, pages 582–591, Providence, RI, May 2003. IEEE.
- Bennet S. Yee. A sanctuary for mobile agents. In *Secure Internet Programming*, number 1603 in Lecture Notes in Computer Science, pages 261–274. Springer-Verlag Inc., New York, NY, USA, 1999.
- Bennet S. Yee. Securely improving performance through speculative predictive remote execution. Presented at the Future Directions in Distributed Computing Workshop, March 2004.
- Xianan Zhang, Dmitrii Zagorodnov, Keith Marzullo, Matti Hiltunen, and Richard Schlichting. Fault-tolerant grid services using primary-backup: Feasibility and performance. In *Cluster 2004: Proceedings of the 2004 IEEE International Conference on Cluster Computing*, 2004.

References

[1] Martín Abadi and Joan Feigenbaum. Secure circuit evaluation. *Journal of Cryptography*, 2(1):1–12, 1990.

[2] Mike Accetta, Robert V. Baron, William Bolosky, David B. Golub, Richard F. Rashid, Avadis Tevanian, Jr., and Michael Wayne Young. Mach: A new kernel foundation for Unix development. In *Proceedings of Summer USENIX*, July 1986.

[3] William Aiello, Sandeep Bhatt, Rafail Ostrovsky, and S. Raj. Rajagopalan. Fast verification of any remote procedure call: Short witness-indistinguishable one-round proofs for NP. In U. Montanarai et al., editor, *Proceedings of International Colloquium on Automata, Languages and Programming*, pages 463–474, 2000.

[4] Mihir Bellare and Bennet Yee. Forward-security in private-key cryptography. In M. Joye, editor, *Topics in Cryptography—CT-RSA '03*, volume 2612 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.

[5] Carl Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian Thomas, and Tatu Ylonen. SPKI certificate theory. Internet RFC 2693, September 1999.

[6] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, and Dan Boneh. Terra: A virtual machine-based platform for trusted computing. In *Proceedings of the 19th Symposium on Operating System Principles(SOSP 2003)*, October 2003.

[7] Jonathan T. Giffin, Somesh Jha, and Barton P. Miller. Detecting manipulated remote call streams. In *Proceedings of the Eleventh USENIX Security Symposium*, pages 61–79. USENIX Association, 2002.

[8] Matthew Hohlfeld, Aditya Ojha, and Bennet Yee. Security in the Sanctuary system. Technical Report CS2002-0731, Computer Science and Engineering Department, University of California at San Diego, December 2002.

[9] Matthew Hohlfeld and Bennet S. Yee. How to migrate agents. Technical Report CS98-588, Computer Science and Engineering Department, University of California at San Diego, La Jolla, CA, August 1998.

[10] Kjetil Jacobsen, Xianan Zhang, and Keith Marzullo. Group membership and wide-area master-worker computations. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS'03)*, pages 570–581, Providence, RI, May 2003. IEEE.

[11] Stephen Thomas Kent. *Protecting Externally Supplied Software in Small Computers.* PhD thesis, Massachusetts Institute of Technology, September 1980.

[12] Rahul S. Lahoti. Continuations in Java. Master's thesis, University of California, San Diego, July 2003.

[13] Aditya Ojha. Tamper-evident mobile agents. Master's thesis, Computer Science and Engineering, University of California, San Diego, 2003.

[14] Scott Owen O'Neil. Secure mobile agent aware remote procedure calls in S^2PRE. Master's thesis, University of California, San Diego, 2004.

[15] Mudumbai Ranganathan, Anurag Acharya, Shamik D. Sharma, and Joel Saltz. Network-aware mobile programs. In *Proceedings of the Usenix 1997 Annual Technical Conference*. Usenix, 1997.

[16] Ronald L. Rivest and Butler Lampson. SDSI–a simple distributed security infrastructure. (see SDSI web page at `http://theory.lcs.mit.edu/~cis/sdsi.html`).

[17] Yekaterina Yevgenyevna Tsipenyuk. Detecting external agent replay and state modification attacks. Master's thesis, University of California, San Diego, 2004.

[18] U. S. Department of Defense, Computer Security Center. Trusted computer system evaluation criteria, December 1985.

[19] U. S. National Institute of Standards and Technology. (draft) federal information processing standards publication 140-2: Security requirements for cryptographic modules.

[20] Poornaprajna Venkatraj Udupi. Security attribute certification infrastructure (SACI). Master's thesis, University of California, San Diego, 2003.

[21] Juliana Wong. A secure network IPC system for mobile agent systems. Master's thesis, University of California, San Diego, 2004.

[22] Bennet Yee. Monotonicity and partial results protection for mobile agents. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS'03)*, pages 582–591, Providence, RI, May 2003. IEEE.

[23] Bennet S. Yee. A sanctuary for mobile agents. In *Secure Internet Programming*, number 1603 in Lecture Notes in Computer Science, pages 261–274. Springer-Verlag Inc., New York, NY, USA, 1999.

[24] Bennet S. Yee. Securely improving performance through speculative predictive remote execution. Presented at the Future Directions in Distributed Computing Workshop, March 2004.

[25] Bennet S. Yee and Doug Tygar. Secure coprocessors in electronic commerce applications. In *Proceedings of the First USENIX Workshop on Electronic Commerce*, New York, New York, July 1995.

[26] Xianan Zhang, Dmitrii Zagorodnov, Keith Marzullo, Matti Hiltunen, and Richard Schlichting. Fault-tolerant grid services using primary-backup: Feasibility and performance. In *Cluster 2004: Proceedings of the 2004 IEEE International Conference on Cluster Computing*, 2004.